Sección de Matemáticas
Universidad de La Laguna

María Jesús Álvarez Rodríguez

Programación por Metas

Goal Programming

Trabajo Fin de Grado
Grado en Matemáticas
La Laguna, Junio de 2019

Dirigido por
Carlos González Martín

Carlos González Martín
Departamento de Matemáticas, Estadística e Investigación Operativa
Universidad de La Laguna
38200 La Laguna, Tenerife

Agradecimientos

Me gustaría agradecer a mi tutor Carlos González Martín, su apoyo, comprensión y dedicación para realizar este trabajo.

También quiero dar las gracias a mi familia, amigos y a todas aquellas personas que me han apoyado y me han ayudado en aquellos momentos que más lo he necesitado, dandome la fuerza necesaria para continuar luchando en lo que creo y no rendirme.

María Jesús Álvarez Rodríguez
La Laguna, 11 de junio de 2019

Resumen · Abstract

Resumen

En este trabajo, dentro de la Programación Multiobjetivo, nos centraremos en el estudio de la Programación por Metas. Después de hacer un recorrido por los aspectos históricos, presentamos el modelo general y modelizamos algunos ejemplos. También presentamos métodos generales que nos permitirán resolver los problemas de Programación por Metas. Finalmente, resolveremos los problemas propuestos con la ayuda de software disponible para Proramación Lineal.

Palabras clave: *Optimización Multicriterio – Programación por Metas – Programación Lineal – Complementos de Optimización de hojas de cálculo – R*

Abstract

This project will focus on the study of Goal Programming within Multiobjective Programming. After having outlined the historical aspects, a general model is presented and some examples are modelled. General methods are also presented which will allow us to solve the problems of Goal Programming. Finally, the proposed problems will be solved with the software available for Linear Programming.

Keywords: *Multi-criteria Optimization – Goal Programming – Linear Programming – Spreadsheets Optimization add-ons – R*

Contenido

Introducción

En diferentes contextos que involucran a las personas, y en el desarrollo de sus quehaceres vitales, surgen situaciones problemáticas que exigen soluciones con atributos de optimalidad. Algunos de estos casos se pueden formalizar como modelos matemáticos cuyo estudio es responsabilidad de la Programación Matemática u Optimización, bajo el paraguas de la Investigación Operativa.

La necesaria optimalidad puede incluir, simultáneamente, más de un criterio. Esta exigencia sintoniza propiamente con las formas racionales de tomar decisiones y nos sitúa en el campo de conocimiento de la Decisión Multicriterio o, más concretamente, de la Programación Multiobjetivo.

En este contexto, es común que existan conflictos entre los diferentes objetivos y esta realidad impone nuevas ideas en relación con las establecidas para la Optimización Uniobjetivo.

Veamos un ejemplo.
En una empresa de transporte, se pretende minimizar el coste de transportar la mercancía a sus destinos. Se nos presenta, por tanto, un problema en el que tenemos que satisfacer un único objetivo.

Modifiquemos el problema, y pensemos que no sólo queremos minimizar el coste de transportar la mercancía, sino que también queremos minimizar la distancia que realiza el repartidor para completar los envíos, el tiempo de entrega, el tiempo en el que el repartidor se encuentra en cola a lo largo de su recorrido, el tiempo de entrega de los pedidos urgentes, etc.
En este caso nos enfrentamos ante un problema multiobjetivo. La búsqueda de soluciones posibles se ha de realizar teniendo presente esta realidad.

En algunos casos relevantes, los criterios que gobiernan el proceso de optimización imponen la superación de determinadas metas. Cuando se especifican metas o niveles de satisfacción que deben ser alcanzados, nos ubicamos en lo que se conoce como Programación por Metas.

Dedicaremos el presente trabajo a estudiar este tipo de problemas.
En el capítulo 1 estudiaremos los conceptos básicos, el modelo general y algunos ejemplos de problemas de Programación Multiobjetivo.

En el capítulo 2 nos centraremos en estudiar la Programación por Metas. Haremos un recorrido por su historia, los conceptos básicos, el modelo general y veremos algunos ejemplos.

En el capítulo 3 veremos distintos métodos generales que nos permitan resolver los problemas de Programación por Metas. Los métodos que estudiaremos son: el método de metas lexicográficas, el método de metas ponderadas y el método MINMAX.

Para finalizar, en el capítulo 4 usaremos distintas herramientas computacionales que nos permitan resolver los ejemplos propuestos en capítulos anteriores.
Además, se completa este trabajo con bibliografía empleada para el desarrollo del mismo y el poster.

1

Programación Multiobjetivo. Terminología y conceptos básicos

1.1. Introducción

En este capítulo presentaremos los conceptos básicos de la Programación Multiobjetivo y el modelo general. Además, incluiremos un ejemplo de problema bicriterio.

1.2. Conceptos básicos

En general, en los procesos de Investigación Operativa, dos figuras fundamentales en el estudio y resolución de los problemas multiobjetivo son el decisor y el analista.

El **decisor** es la persona (grupo de personas, entidad, ...) que detecta los problemas y es responsable de su resolución.
El **analista** (persona, gabinete, entidad, ...) representa a la parte técnica especializada y es conocedor de herramientas específicas para encontrar soluciones. Debe trabajar en relación estrecha con el decisor.

Relacionamos a continuación algunos elementos básicos necesarios para una formalización adecuada de estos problemas.

Definición 1.1. *Se denominan **objetos** a la unidades o ítems elementales (físicas, conceptuales, ...) involucradas en la situación problemática en estudio. Se pueden denominar alternativas.*

Definición 1.2. *Se denominan **atributos** a las características medidas sobre los objetos. Son, esencialmente, variables.*
En el caso de tener un conjunto finito de alternativas, la consideración conjunta de objetos y atributos aporta la siguiente tabla:

	X_1	...	X_j	...	X_n
o_1	x_{11}	...	x_{1j}	...	x_{1n}
...	...		...		...
o_i	x_{i1}	...	x_{ij}	...	x_{1n}
...	...		...		...
o_m	x_{m1}	...	x_{mj}	...	x_{mn}

Definición 1.3. *El **espacio de decisiones** viene definido por los vectores de valores que alcanzan los diferentes atributos en cada objeto. Se suele denominar por X o por S.*

Definición 1.4. *Los **objetivos** son funciones, de valor real, establecidas sobre los niveles que alcanzan los atributos.*

Definición 1.5. *Las **metas** son cotas asociadas a atributos u objetivos cuya aproximación o superación sea esencial para el problema.*

Definición 1.6. *Los **criterios** son los que establecen las pautas que articulan el marco global de las búsquedas de soluciones. Incluyen objetivos, metas y atributos relevantes para los procesos de resolución.*

Definición 1.7. *El **espacio objetivo** está determinado por el conjunto* $f(S) = \{(f_1(x), ..., f_p(x))|x \in S\}$

Utilizando los conceptos anteriores podemos introducir el siguiente modelo general de Programación Multiobjetivo.

1.3. Modelo general de la Programación Multiobjetivo

Dados, $S \subseteq R^N, S \neq \phi$, $f_j : S \to R$, $j \in \{1, ..., p\}$, el problema de optimización multiobjetivo se plantea en los términos:

$$\begin{aligned} &min\ (f_1(x), ..., f_p(x)) \\ &s.a: \quad x \in S \end{aligned} \tag{1.1}$$

Para $j \in \{1, ..., p\}$, supongamos el problema uniobjetivo:

$$\begin{aligned} &min \quad f_j(x) \\ &s.a: \ x \in S \end{aligned}$$

tiene solución óptima $\bar{x}^j$ con valor óptimo f_j^*.

Definición 1.8. *Una* ***solución ideal*** *es cualquier solución factible x^* del problema multiobjetivo en la que $(f_1(x^*), ..., f_p(x^*)) = (f_1^*, ..., f_p^*)$.*
Al vector $(f_1^*, ..., f_p^*)$ se le denomina ***punto ideal o utopía****.*

En general, las soluciones ideales son no factibles. La, en general, no factibilidad del óptimo absoluto impone la búsqueda de soluciones optimales y conduce a la siguiente definición.

Definición 1.9. *$\bar{x} \in X$ es una* ***solución eficiente*** *sí, y sólo si, $\nexists x' \in S$ tal que $f_j(x') \leq f_j(\bar{x}), \forall j \in \{1, ..., p\}$ y $f_{j_0}(x') < f_{j_0}(\bar{x})$ para algún $j_0 \in \{1, ..., p\}$.*
Estas soluciones también se denominan no dominadas u óptimas de Pareto.

Definición 1.10. *$\bar{x} \in X$ es una* ***solución débilmente eficiente*** *sí, y sólo si, $\nexists x' \in S$ tal que $f_j(x') < f_j(\bar{x}), \forall j \in \{1, ..., p\}$*

Cualquier solución eficiente es débilmente eficiente. Lo contrario no es cierto.
En general, el conjunto de soluciones eficientes es muy numeroso.

Definición 1.11. *Una* ***solución preferida*** *es la que optimiza las preferencias del decisor. La coherencia impone que la solución preferida tenga que elegirse entre las soluciones eficientes.*

Definición 1.12. *Si las preferencias del decisor pueden establecerse a través de una función sobre los valores que alcanzan los objetivos, tenemos una* ***función de utilidad****.*

Obviamente, si se conoce la función de utilidad, encontrar soluciones al problema planteado se convierte en resolver un problema uniobjetivo. Las soluciones preferidas del decisor deben ser soluciones factibles (eficientes) en las que la función de utilidad alcance valor máximo.

Sin embargo, aunque el decisor sea capaz de valorar su utilidad en las diferentes soluciones posibles, es difícil, en general, que se puedan plasmar sus preferencias en una función. Más fácil le resultaría, en algunos casos, establecer niveles de conformidad o de satisfacción para cada objetivo (criterio, atributos, ...) o estar dispuesto a aceptar "pérdidas" respecto a niveles ideales.

Definición 1.13. *Se llaman* ***soluciones satisfactorias*** *a las que superan determinados niveles (que expresan la satisfacción del decisor) asociados a los diferentes objetivos (atributos).*

En este trabajo nos centraremos, a partir del capítulo 2, en la búsqueda de soluciones satisfactorias que superen determinadas metas específicadas por el decisor.

Definición 1.14. *Se llaman* ***soluciones de compromiso*** *a las soluciones factibles en las que las desviaciones de los niveles de las funciones objetivos (atributos) respecto al nivel ideal son pérdidas pueden ser aceptadas por el decisor.*

Lo anterior indica que la búsqueda de soluciones al problema de optimización multiobjetivo debe incluir procesos de generación de soluciones eficientes que, o bien siguen criterios que incorporan las preferencias del decisor, o, cuando es posible, construyen todo el conjunto de dichas soluciones. En este último caso, en una etapa posterior, la selección de la solución preferida se hará en el citado conjunto.

1.4. Ejemplo

Veamos un ejemplo de problema de Programación Lineal biobjetivo, que aparece en [5]

Ejemplo 1.1
En una fábrica se pretende producir grandes cantidades de dos sustancias químicas, las cuales denominaremos A y B. La empresa tiene como objetivo maximizar el beneficio obtenido por la fabricación de ambas sustancias. Pero, además, la empresa tiene otro objetivo a satisfacer y es minimizar la contaminación emitida en la fabricación de ambas sustancias. Tenemos, por tanto, un problema biobjetivo.

Las variables de decisión de este problema son:
x_1= cantidad producida de A
x_2= cantidad producida de B

y las funciones objetivo que se plantean son:

$$max\ z_1(x) = 2x_1 + 3x_2$$
$$min\ z_2^o(x) = 4x_1 + 2x_2$$

Donde z_1 hace referencia al beneficio obtenido y z_2^o hace referencia a la contaminación emitida en la producción de las sustancias.
Además, multiplicamos z_2^o por -1 (para tener de esta manera ambas funciones objetivo de máximo), de esta forma, $z_2 = -z_2^o$.

Las restricciones que se tienen son:

$$x_1 + 3x_2 \leq 24$$
$$2x_1 + x_2 \leq 18$$
$$x_1 + x_2 \leq 10$$
$$x_1, x_2 \geq 0$$

Estas restricciones limitan la cantidad de materia prima para la producción, los tiempos de trabajo en la fabricación, imponen un control en la calidad y la condición de no negatividad de las variables de decisión, respectivamente.

El problema queda, por tanto, de la siguiente manera:

$$\begin{aligned} max \;\; & z = (z_1(x), z_2(x)) \\ s.a: \;\; & x_1 + 3x_2 \leq 24 \\ & 2x_1 + x_2 \leq 18 \\ & x_1 + x_2 \leq 10 \\ & x_1, x_2 \geq 0 \end{aligned}$$

Realicemos a continuación un estudio gráfico de este problema.
En primer lugar, representamos la región factible denotada por S en la imagen a) , junto con las funciones z_1 *y* z_2^o. Además indicamos, con el sentido de las flechas, si la función es de mínimo o máximo. Nos percatamos de que en este caso los objetivos entran en conflicto. Esto se debe tener en cuenta para la posterior resolución del problema.

A continuación, proyectamos la imagen de S en $z_1(x)$ y $z_2(x)$.
Vemos que el conjunto de soluciones eficientes es el que está formado por $\overline{OA}$ *y* $\overline{AB}$. Y es el desisor en este caso quien elegirá la mejor solución según sus intereses.

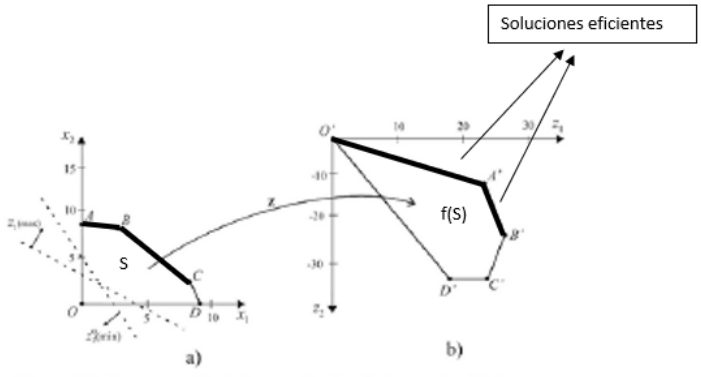


Figura 1.1. Espacio de decisiones y espacio objetivo

Vemos que las soluciones eficientes son $\overline{O'A'}$ *y* $\overline{A'B'}$.

Además en la siguiente tabla se muestra la imagen de los puntos extremos de S proyectados en f(S).

Punto extremo en S	(x_1, x_2)	(z_1, z_2)	Punto extremo en f(S)
O	(0,0)	(0,0)	O'
A	(0,8)	(24,-16)	A'
B	(3,7)	(27,-26)	B'
C	(8,2)	(22,-36)	C'
D	(9,0)	(18,-36)	D'

2

Problemas de Programación por Metas.

2.1. Introducción

En este capítulo introduciremos los conceptos básicos relacionados con la Programación por Metas, haremos un breve resumen de los aspectos históricos relacionados más destacados, veremos el modelo general para este tipo de problemas e introduciremos algunos ejemplos que resolveremos más adelante.

2.2. Aspectos históricos

Tal y como se indica en [7], la Programación por Metas es una metodología útil para la resolución de determinados problemas de Programación Multiobjetivo. Su introducción inicial se puede situar en 1955 con la publicación de un artículo de Charnes, Cooper y Ferguson. Esta publicación tuvo como motivo desarrollar un método para poder determinar los salarios de los ejecutivos de la compañía General Electric.

Esta metodología surgió por la necesidad de considerar nuevas restricciones con la contemplación de distintos niveles que era necesario satisfacer. Por tanto, Charnes, Cooper y Ferguson desarrollaron un modelo con restricciones donde se minimizaba la suma de las desviaciones absolutas respecto a los niveles ya establecidos. Además, tuvieron que introducir variables de desviación positivas y negativas debido a que la desviación absoluta no era lineal.

Pese a que el citado trabajo representa el nacimiento de la Programación por Metas, fue en el libro "Management Models and Industrial Applications of Linear Programming", por Charnes y Cooper, la primera vez que se utilizó el término de Programación por Metas. Además, cabe destacar que Charnes y Cooper no hablaron en su libro de la Programación por Metas como un método para la

resolución de problemas multiobjetivo, sino que la introdujeron como un modelo alternativo a la estimación de parámetros por el método de mínimos cuadrados.

Fueron muchos los que entre los años 60 y 70 desarrollaron y trabajaron aspectos de la Programación por Metas. Cabe destacar a Ignizio en el año 1963 que fue capaz de adaptar el término de Programación por Metas para obtener soluciones en un problema que conllevaba la organización del sistema de antenas Saturno/Apolo. La aplicación de la Programación por Metas era consecuencia de que en dicho problema se encontraban múltiples metas, además de funciones no lineales y variables enteras.

También en 1963, Charnes et al. aplicaron esta modelización a problemas financieros y contables. Más tarde, en 1968 lo extendieron a casos de planificación de publicidad. Además, en 1972, Lee y, en 1976, Ignizio publicaron dos libros en los que se perfilaban y ampliaban los aspectos conceptuales y metodológicos de la Programación por Metas. Estos libros sirvieron de gran impulso para trabajar en numerosas aplicaciones.

Algunas de las áreas en las que se ha aplicado la Programación por Metas son: control de calidad, programación económica, planificación y gestión de recursos agrarios, recursos sanitarios, publicidad,...

Cabe destacar la aplicación de la Programación por Metas a problemas reales en los trabajos de Carlos Romero. Y es por ello que en este trabajo se hace indispensable realizar consultas bibliográficas a algunos de sus escritos como por ejemplo: [6] y [7].

2.3. Planteamiento del modelo general

Dados, $S \subseteq R^N, S \neq \phi,\ f_j : S \to R,\ j \in \{1, ..., p\}$, supongamos que manejamos el modelo multicriterio presentado previamente:

$$\begin{aligned} &min\ (f_1(x), ..., f_p(x)) \\ &s.a: \quad x \in S \end{aligned} \tag{2.1}$$

Entenderemos que los objetivos establecidos incorporan los criterios de optimización que especifica el decisor.
Supongamos, además, que esos criterios introducen metas para los objetivos; es decir: $\forall j \in \{1, ..., p\},\ m_j$ es la meta asociada al objetivo j.

Como, en principio, el valor de $f_j(x)$ puede ser menor o igual o mayor o igual que m_j, podemos considerar desviaciones por defecto o por exceso y escribir, $\forall j \in \{1, ..., p\}$:

$$f_j(x) + d_j^- - d_j^+ = m_j,\ d_j^- \geq 0,\ d_j^+ \geq 0$$

2.4. Conceptos básicos

A continuación, introduciremos diversos conceptos elementales (ver, por ejemplo [2] y [7]), que nos ayuden a introducir las características esenciales de los problemas de Programación por Metas.

Definición 2.1. *Se denomina* ***variable de desviación negativa*** d_i^- *a la variable que mide la diferencia que falta para que la meta alcance el nivel de aspiración.*

Definición 2.2. *Se denomina* ***variable de desviación positiva*** d_i^+ *a la variable que mide el exceso que hay desde el nivel de aspiración y la meta.*

Dentro del proceso de optimización, interesará que $f_j(x) \leq m_j$, o, por el contrario, que $f_j(x) \geq m_j$. Esto implica, necesariamente que $d_j^- = 0$ ó $d_j^+ = 0$.

Definición 2.3. *Se denomina* ***variable de desviación no deseada*** *a la variable que queremos que tenga el menor valor posible.*

Cuando nuestro criterio es superar una meta, nuestra variable de desviación no deseada será la variable de desviación negativa. Por el contrario, cuando nuestro criterio sea no superar una meta, nuestra variable de desviación no deseada será la variable de desviación positiva.

Definición 2.4. *En los problemas de programación por metas el vector* $a = (a_1, a_2, ..., a_p)$, *donde* $a_i = g_i(d_i^+, d_i^-), i = 1, 2, ..., p$, *se denomina* ***vector de logro****. Cada* a_i *puede ser lineal o no lineal.*

La introducción del vector de logro permite escribir el problema de Programación por Metas como un nuevo problema de Programación Multiobjetivo:

$$\begin{aligned} &min\ \ (a_1, ..., a_p) \\ &s.a: f_j(x) + d_j^- - d_j^+ = m_j,\ j = 1, ..., p \\ &\quad x \in S \\ &\quad d_j^-, d_j^+ \geq 0,\ j = 1, ..., p. \end{aligned} \tag{2.2}$$

La condición de complementariedad de las desviaciones por exceso y por defecto asociadas a cada meta, es una restricción coherente con el planteamiento del modelo: no pueden obtenerse soluciones que, a la vez, superen y estén por debajo de una meta específica. Generalmente esta condición se maneja de forma implícita en los distintos procesos de resolución.

Definición 2.5. *En algunos de los procesos de resolución de problemas de Programación por Metas se suelen combinar las distintos a_i en una función que se denomina* ***función de logro****. Es, propiamente, una función de utilidad sobre los a_i que permiten convertir el problema planteado en un problema uniobjetivo, como el que se muestra a continuación:*

$$\begin{aligned} & opt\ F.L.(a_1, ..., a_p) \\ & s.a: f_j(x) + d_j^- - d_j^+ = m_j,\ j = 1, ..., p \\ & \quad x \in S \\ & \quad d_j^-, d_j^+ \geq 0,\ j = 1, ..., p. \end{aligned} \tag{2.3}$$

con opt entendemos que la función objetivo puede ser de máximo o mínimo.

Resolveremos casos en los que para mayor facilidad de trabajo, la función de logro es lineal.

Es deseable que las características de las a_i y de la función de logro, hagan posible que las soluciones encontradas para (2.2) y, por extensión, para (2.3) tengan propiedades de eficiencia.

2.5. Ejemplos

Ejemplo 2.1

Modifiquemos ahora el **Ejemplo 1.1**, tomado de [5] y planteémoslo ahora como un problema de Programación por Metas.
Como metas se tomarán: $m_1 = 2x_1 + 3x_2 = 28$ *y* $m_2 = 4x_1 + 2x_2 = 24$.

Tenemos por tanto el siguiente problema:

$$\begin{aligned} & min\ d_1^- + d_1^+ + d_2^- + d_2^+ \\ & s.a: x_1 + 3x_2 \leq 24 \\ & \quad 2x_1 + x_2 \leq 18 \\ & \quad x_1 + x_2 \leq 10 \\ & \quad 2x_1 + 3x_2 - d_1^+ + d_1^- = 28 \\ & \quad 4x_1 + 2x_2 - d_2^+ + d_2^- = 24 \\ & \quad x_1, x_2 \geq 0\ y\ d_i^+, d_i^- \geq 0,\ i = 1, 2 \end{aligned}$$

donde $a_1 = d_1^- + d_1^+$ y $a_2 = d_2^- + d_2^+$.

Ejemplo 2.2
Usamos un ejemplo de [1].
La empresa Harrison Electric Company, en Chicago, fabrica candelabros y ventiladores de techo de estilo antiguo. Dicha fábrica desea maximizar los beneficios obtenidos con la producción de los citados productos.

Para la fabricación de los ventiladores y candelabros se realizan dos pasos: el cableado eléctrico y el ensamble. Para cablear un candelabro se necesitan 2 horas y para ensamblarlo 6. Además, para cablear cada ventilador se necesitan 3 horas y 5 horas para ensamblarlo. Cabe destacar que sólo se dispone de 12 horas para realizar el cableado y 30 horas para el ensamble.

En la venta de cada candelabro se obtiene 7 dolares y en la venta cada ventilador 6 dolares.

Modelizamos el problema que se nos plantea.

$$\begin{aligned} max\ & 7x_1 + 6x_2 \\ s.a : & 2x_1 + 3x_2 \leq 12 \\ & 6x_1 + 5x_2 \leq 30 \\ & x_1, x_2 \geq 0 \end{aligned}$$

siendo las variables de decisión:
x_1= candelabros fabricados
x_2= ventiladores fabricados

Pasado un tiempo, la empresa decide mudarse y en este momento se valora que el objetivo anterior no es del todo realista. Por tanto, la fábrica considera que obtener un beneficio de 30 dolares por día, sí es un objetivo que puede ayudar a la empresa de manera real.
Estamos ahora frente a un problema de programación por metas con las restricciones de tiempo de producción que anteriormente se tenían.

Definimos las variables de desviación:
d_1^- = beneficio de venta inferior a 30 dolares
d_1^+ = beneficio de venta superior a 30 dolares

Tenemos por tanto el siguiente problema:

$$\begin{aligned} & min\ d_1^- + d_1^+ \\ & s.a: 7x_1 + 6x_2 + d_1^- - d_1^+ = 30 \\ & \quad 2x_1 + 3x_2 \leq 12 \\ & \quad 6x_1 + 5x_2 \leq 30 \\ & \quad x_1, x_2, d_1^-, d_1^+ \geq 0 \end{aligned} \tag{2.4}$$

Nuevamente la empresa considera que se puede mejorar la situación. En esta ocasión, la administración de la fábrica se plantea alcanzar las siguientes metas, cada una con igual prioridad:
Meta 1: Obtener un beneficio de 30 dolares cada día
Meta 2: Utilizar todas las horas disponibles para el trabajo de cableado
Meta 3: Evitar que en el departamento de ensamble se superen las horas establecidas
Meta 4: Producir como mínimo 7 ventiladores de techo

Considerando estas metas tenemos las variables de desviación siguientes:
d_1^- y d_1^+: variables de desviación relacionadas con el mínimo beneficio a lograr
d_2^- y d_2^+: variables de desviación relacionadas con el tiempo disponible del departamento de cableado
d_3^- y d_3^+: variables de desviación relacionadas con el tiempo excesivo de trabajo del departamento de cableado
d_4^- y d_4^+: variables de desviación relacionadas con el mínimo de ventiladores de techo que se producen

Teniendo en cuenta las metas que se han establecido, las variables de desviación $d_1^+,\ d_2^+,\ d_3^-$ y d_4^+ se pueden eliminar de la función objetivo.

El problema queda entonces de la siguiente manera:

$$\begin{aligned} & min\ d_1^- + d_2^- + d_3^+ + d_4^- \\ & s.a: 7x_1 + 6x_2 + d_1^- - d_1^+ = 30 \\ & \quad 2x_1 + 3x_2 + d_2^- - d_2^+ = 12 \\ & \quad 6x_1 + 5x_2 + d_3^- - d_3^+ = 30 \\ & \quad x_2 + d_4^- - d_4^+ = 7 \\ & \quad x_i, d_j^-, d_j^+ \geq 0,\ \forall i = 1, 2\ y\ \forall j = 1, 2, 3, 4 \end{aligned} \tag{2.5}$$

donde $a_1 = d_1^-,\ a_2 = d_2^-,\ a_3 = d_3^+$ y $a_4 = d_4^-$.

Ejemplo 2.3
A continuación, estudiamos otro ejemplo de Programación por Metas, esta vez obtenido de [4].

Una empresa produce semanalmente dos tipos de productos A y B. Sabemos que para fabricar el producto A se necesitan 4 horas y para el producto B, 3 horas. Además sabemos que se obtiene 100 euros y 150 euros respectivamente como beneficio del producto A y B.

Para la producción de estos productos se necesita de cierto material, el cual se debe comprar como mínimo 50 litros por semana. En concreto, para producir A se necesitan 2 litros y para producir B 1 litro del material anteriormente mencionado.Además la empresa tiene problemas con los tiempos disponibles para el uso de la maquinaria y por ello en una semana sólo se pueden fabricar 75 productos.

En nuestro problema tenemos dos variables de decisión:
x_1: cantidad de A que se produce semanalmente
x_2: cantidad de B que se produce semanalmente

Y las restricciones que debemos tener en cuenta son:

$$2x_1 + x_2 \geq 50$$
$$x_1 + x_2 \leq 75$$

Además dicha empresa desea satisfacer las siguientes metas:
Meta 1: No utilizar más de 120 horas de trabajo en la fabricación de los productos A y B.
Meta 2: La empresa quiere obtener un beneficio de al menos 7000 euros.
Meta 3: Fabricar al menos 40 productos tipo A.
Meta 4: Fabricar al menos 40 productos tipo B.

Se tiene con todo ello el siguiente problema:

$$
\begin{aligned}
& Min\ d_1^+ + d_2^- + d_3^- + d_4^- \\
& s.a: 4x_1 + 3x_2 + d_1^- - d_1^+ = 120 \\
& \quad 100x_1 + 150x_2 + d_2^- - d_2^+ = 7000 \\
& \quad x_1 + d_3^- - d_3^+ = 40 \\
& \quad x_2 + d_4^- - d_4^+ = 40 \\
& \quad 2x_1 + x_2 \geq 50 \\
& \quad x_1 + x_2 \leq 75 \\
& \quad x_1, x_2 \geq 0, d_j^-, d_j^+ \geq 0, \forall j = 1, 2, 3, 4, 5, 6
\end{aligned}
\tag{2.6}
$$

donde $a_1 = d_1^+,\ a_2 = d_2^-,\ a_3 = d_3^-,\ a_4 = d_4^-$.

Cabe resaltar que en todos los ejemplos vistos, todas las metas tienen igual prioridad. En el capítulo siguiente veremos otros problemas de Programación por Metas en los que las metas que han satisfacerse tienen prioridades asignadadas, distintas entre sí.

3

Métodos de resolución de problemas de Programación por Metas

3.1. Introducción

En este capítulo estudiaremos distintos métodos para la resolución de los problemas de Programación por Metas, en los que se hacen distintos tratamientos de las variables de desviación. En las aplicaciones, como ya hemos indicado, nos restringimos al caso lineal. De esta forma, distinguimos entre los siguientes métodos:

- Métodos de metas lexicográficas.
- Métodos de metas ponderadas.
- Métodos MINMAX.

3.2. Métodos de metas lexicográficas

En estos métodos el decisor ordena las metas con orden decreciente de prioridad tal y como se explica en las obras [2] y [6].
Primero se debe alcanzar la meta que tiene mayor prioridad. El problema a resolver es:

$$\begin{aligned} &min\ a_1(d_1^-, d_1^+) \\ &s.a: f_1(x) + d_1^- - d_1^+ = m_1 \\ &\quad x \in S \end{aligned}$$

Y obtenemos x^1 como solución de (3.1).
A continuación, consideramos la segunda meta en orden de prioridad y obtenemos el siguiente problema:

$$\begin{aligned} &min\ a_2(d_2^-, d_2^+) \\ &s.a: f_2(x) + d_2^- - d_2^+ = m_2 \\ &\quad f_1(x) = f(x^1) \\ &\quad x \in S \end{aligned}$$

Y, así, sucesivamente consideraremos el problema i-ésimo:

$$\begin{aligned} & min\ a_i(d_i^-, d_i^+) \\ & s.a: f_i(x) + d_i^- - d_i^+ = m_i \\ & \quad f_j(x) = f(x^j),\ j = 1, ..., i-1 \\ & \quad x \in S \end{aligned}$$

Este problema tendrá solución cuando consideremos las r metas que tenemos en el problema o cuando la solución de alguno de los problemas sea un único punto. Pero existe el riesgo de que los problemas a resolver en el proceso sean no factibles ya que son altamente restringidos.

A continuación, veamos diferentes ejemplos para ilustrar este método.

Ejemplo 3.1
Modifiquemos el **Ejemplo 2.1**, de [5], tomando las metas con las prioridades siguientes:
Prioridad P_1: Respetar las restricciones del ejemplo anterior
Prioridad P_2: Obtener al menos un beneficio de 21 euros
Prioridad P_3: No superar 12 unidades de contaminación

Y obtenemos el siguiente problema:

$$\begin{aligned} & min\ (d_1^+ + d_2^+ + d_3^+,\ d_4^+,\ d_5^+) \\ & s.a: x_1 + 3x_2 - d_1^+ + d_1^- = 24 \\ & \quad 2x_1 + x_2 - d_2^+ + d_2^- = 18 \\ & \quad x_1 + x_2 - d_3^+ + d_3^- = 10 \\ & \quad 2x_1 + 3x_2 - d_4^+ + d_4^- = 21 \\ & \quad 4x_1 + 2x_2 - d_5^+ + d_5^- = 12 \\ & \quad x_i \geq 0,\ i = 1, 2\ \ y\ \ d_j^+, d_j^- \geq 0,\ j = 1, 2, 3, 4, 5 \end{aligned}$$

donde $a_1 = d_1^+ + d_2^+ + d_3^+,\ a_2 = d_4^+\ y\ a_3 = d_5^+$.

Resolvamos el ejemplo teniendo en cuenta, en primer lugar, las metas que tienen la mayor prioridad. Obtenemos la siguiente región factible que se representa en el área sombreada.

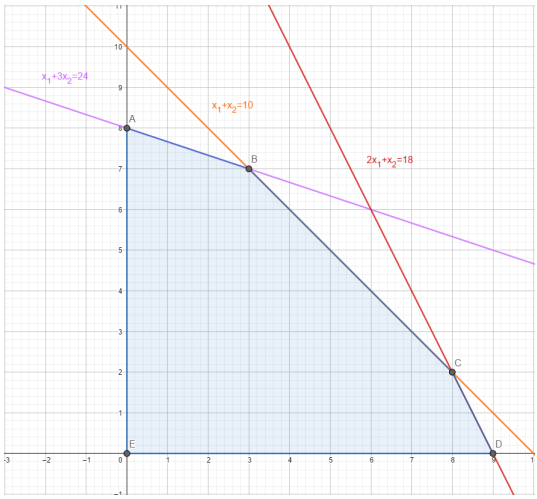


Figura 3.1. Región factible

Ahora consideramos la meta con prioridad segunda y obtenemos la siguiente región factible:

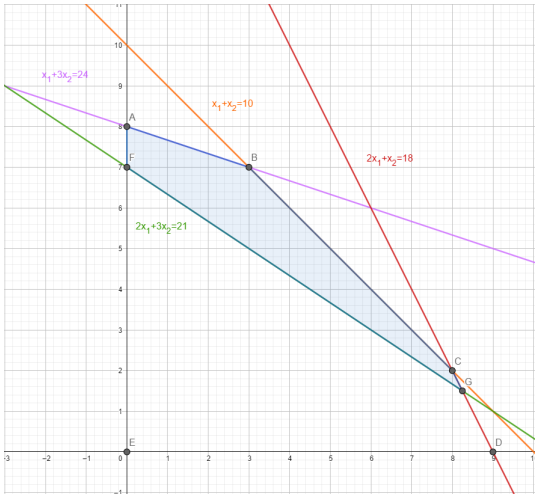


Figura 3.2. Región factible

Por último, resolvemos el problema teniendo en cuenta la meta con prioridad menor y obtenemos:

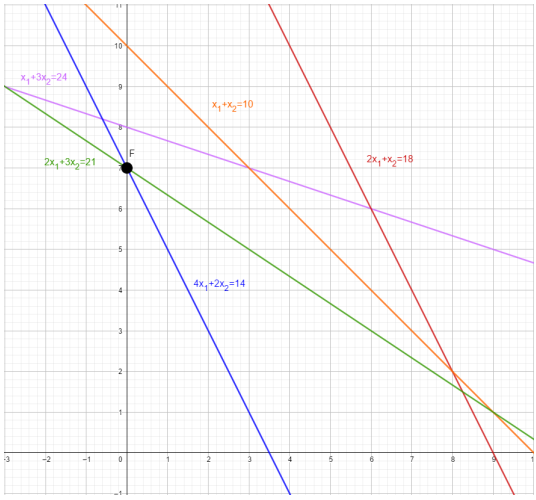


Figura 3.3. Región factible

Nota: se observa que la región factible se reduce al punto (0,7).

Ejemplo 3.2
Partimos del **Ejemplo 2.2**, encontrado en [1] y suponemos ahora que la compañía establece las siguientes prioridades a las respectivas metas:
Prioridad P_1: Obtener como mínimo un beneficio de 30 dolares.
Prioridad P_2: Utilizar todas las horas que están disponibles en el departamento de cableado.
Prioridad P_3: Evitar realizar horas extras en el departamento de ensamblaje.
Prioridad P_4: Fabricar como mínimo siete ventiladores de techo.

Teniendo en cuenta las prioridades asignadas a las metas, obtenemos el problema siguiente:

$$
\begin{aligned}
&min\ (d_1^-,\ d_2^-,\ d_3^+,\ d_4^-)\\
&s.a: 7x_1 + 6x_2 + d_1^- - d_1^+ = 30\\
&\qquad 2x_1 + 3x_2 + d_2^- - d_2^+ = 12\\
&\qquad 6x_1 + 5x_2 + d_3^- - d_3^+ = 30\\
&\qquad x_2 + d_4^- - d_4^+ = 7\\
&\qquad x_i, d_j^-, d_j^+ \geq 0,\ \forall i = 1,2\ y\ \forall j = 1,2,3,4
\end{aligned}
$$

donde $a_1 = d_1^-,\ a_2 = d_2^-,\ a_3 = d_3^+\ y\ a_4 = d_4^-$.
A continuación, pasamos a resolver este problema gráficamente, aplicando el Método Lexicográfico.
Lo primero que hacemos es considerar la meta que tenga mayor prioridad y dibujamos la restricción asociada. Cuando dibujamos la restricción no tenemos en cuenta las variables de desviación. Como nuestro objetivo es minimizar d_1^-, obtenemos como solución el área sombreada.

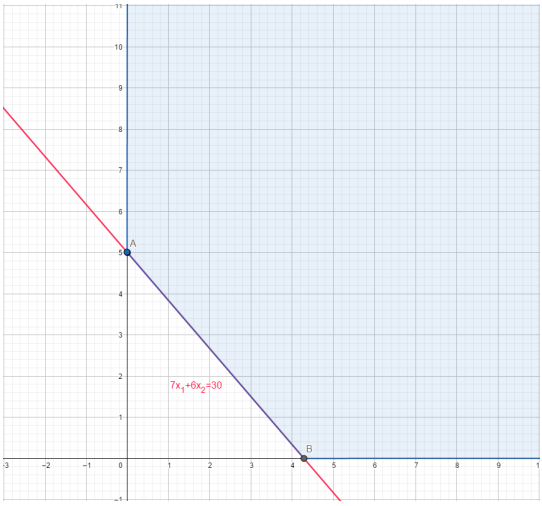


Figura 3.4. Región factible

Luego, buscamos la meta con segunda prioridad que está asociada al control de d_2^- y dibujamos la restricción correspondiente. Debemos tener en cuenta los valores asociados a d_2^+ para evitar tener horas libres en el departamento de cableado. Además debemos hacer compatible esta solución con la obtenida anteriormente. Obtenemos así la figura siguiente:

Figura 3.5. Región factible

A continuación, nos centramos en buscar la meta de prioridad 3. Dibujamos la restricción asociada a la meta que es la que hace referencia a evitar el tiempo extra en el departamento de ensamble. Por tanto, queremos que d_3^+ sea próximo a 0. Debemos buscar soluciones compatibles con las metas anteriores.

Figura 3.6. Región factible

Y por último, tenemos en cuenta la meta con la cuarta prioridad. Dibujamos la restricción asociada a la variable de desviación que utilizamos, en este caso, es la que nos impone producir por lo menos 7 ventiladores de techo. Debemos minimizar d_4^- teniendo en cuenta la región factible anterior.

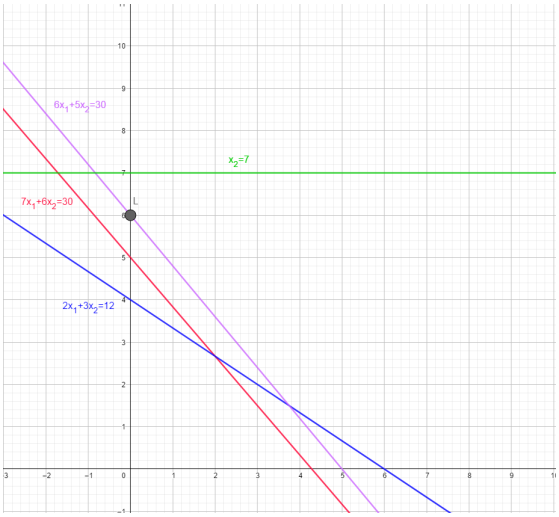


Figura 3.7. Región factible

Por lo que la solución nos queda el punto (0,6).

Ejemplo 3.3
Volvemos ahora al **Ejemplo 2.3** de [4] y supongamos que vamos a dar prioridad a las metas tal y como se indica a continuación:
Prioridad 1: Meta 2 (Obtener al menos 7000 euros de beneficio)
Prioridad 2: Meta 3 y 4 (Fabricar al menos 40 pruductos tipo A y 40 productos tipo B)
Prioridad 3: Meta 1 (No utilizar más de 120 horas de trabajo en la fabricación)

Nuestro función objetivo ahora es:

$$Min\ a = (d_2^-, d_3^- + d_4^-, d_1^+)$$

donde $a_1 = d_2^-,\ a_2 = d_3^- + d_4^-\ y\ a_3 = d_1^+$.

Para resolver este problema, en primer lugar debemos resolver el problema que presentamos a continuación:

$$
\begin{aligned}
& Min\ d_2^- \\
& s.a: 4x_1 + 3x_2 + d_1^- - d_1^+ = 120 \\
& \quad 100x_1 + 150x_2 + d_2^- - d_2^+ = 7000 \\
& \quad x_1 + d_3^- - d_3^+ = 40 \\
& \quad x_2 + d_4^- - d_4^+ = 40 \\
& \quad 2x_1 + x_2 \geq 50 \\
& \quad x_1 + x_2 \leq 75 \\
& \quad x_1, x_2 \geq 0, d_i^-, d_i^+ \geq 0, i = 1, 2, 3, 4
\end{aligned}
$$

De este problema obtenemos que $d_2^- = 0$.
Teniendo en cuenta que $d_2^- = 0$, a continuación, debemos resolver el problema:

$$
\begin{aligned}
& Min\ d_3^- + d_4^- \\
& s.a: 4x_1 + 3x_2 + d_1^- - d_1^+ = 120 \\
& \quad 100x_1 + 150x_2 + d_2^- - d_2^+ = 7000 \\
& \quad x_1 + d_3^- - d_3^+ = 40 \\
& \quad x_2 + d_4^- - d_4^+ = 40 \\
& \quad 2x_1 + x_2 \geq 50 \\
& \quad x_1 + x_2 \leq 75 \\
& \quad d_2^- = 0 \\
& \quad x_1, x_2 \geq 0, d_i^-, d_i^+ \geq 0, i = 1, 2, 3, 4
\end{aligned}
$$

Al resolver este problema obtenemos que las metas a las que se ha asiganado prioridad 2 no pueden ser satisfechas y que el mínimo valor de la función objetivo que podemos alcanzar es 5.

Teniendo en consideración todo ello y la meta que tiene tercera prioridad, resolvemos el siguiente problema:

$$
\begin{aligned}
&Min\ d_1^+ \\
&s.a: 4x_1 + 3x_2 + d_1^- - d_1^+ = 120 \\
&\quad 100x_1 + 150x_2 + d_2^- - d_2^+ = 7000 \\
&\quad x_1 + d_3^- - d_3^+ = 40 \\
&\quad x_2 + d_4^- - d_4^+ = 40 \\
&\quad 2x_1 + x_2 \geq 50 \\
&\quad x_1 + x_2 \leq 75 \\
&\quad d_2^- = 0 \\
&\quad d_3^- + d_4^- = 5 \\
&\quad x_1, x_2 \geq 0, d_i^-, d_i^+ \geq 0, i = 1, 2, 3, 4
\end{aligned}
$$

La resolución de este ejemplo se realizará en el capítulo 4.

3.3. Métodos de metas ponderadas

Este método, estudiado con la ayuda de [2] consiste en asignar pesos a las a_i y considerar que la función de logro viene determinada por:

$$F.L.(a_1, ..., a_p) = G(a_i(w_i^- d_i^-, w_i^+ d_i^+))$$

donde G es una función adecuada y w_i^- y w_i^+ son valores mayores o iguales que cero.

Por tanto, el problema a resolver es:

$$
\begin{aligned}
&opt\ G(a_i w_i^- d_i^-, w_i^+ d_i^+) \\
&s.a: f_i(x) + d_i^- - d_i^+ = m_i \\
&\quad d_i^+, d_i^- \geq 0 \\
&\quad w_i^+, w_i^- \geq 0 \\
&\quad x \in S \\
&\quad \forall\ i = 1, 2, .., p
\end{aligned}
\tag{3.1}
$$

Es importante resaltar que nos encontramos dos dificultades:

En primer lugar, no tiene sentido sumar variables de desviación si estas están medidas en distintas unidades. Y, además, cabe destacar que si los niveles de aspiración son muy distintos, el proceso de resolución tendería a buscar entre las soluciones que cumplan aquellas metas que posean un nivel de aspiración mayor.

Estas dificultades las obviaremos si tomamos, de forma adecuada, las variables de desviación como porcentajes.

Frecuentemente, G es una función lineal, como vemos en los siguientes ejemplos.

Ejemplo 3.4
Partimos del **Ejemplo 1.1** de [5] y añadimos como metas: $m_1 = 2x_1 + 3x_2 = 26$ *y* $m_2 = 4x_1 + 2x_2 = 20$. Además, damos pesos a los distintas variables de desviación como se sigue:

$$w_1^+ = 4,\ w_1^- = 1,\ w_2^+ = 1 \text{ y } w_2^- = 2.$$

Obtenemos así el siguiente problema:

$$\begin{aligned} & min\ 4d_1^+ + d_1^- + d_2^+ + 2d_2^- \\ & s.a: x_1 + 3x_2 \leq 24 \\ & \qquad 2x_1 + x_2 \leq 18 \\ & \qquad x_1 + x_2 \leq 10 \\ & \qquad 2x_1 + 3x_2 - d_1^+ + d_1^- = 26 \\ & \qquad 4x_1 + 2x_2 - d_2^+ + d_2^- = 20 \\ & \qquad x_i, d_i^-, d_i^+ \geq 0, i = 1, 2 \end{aligned}$$

donde $a_1 = 4d_1^+ + d_1^-$ *y* $a_2 = d_2^+ + 2d_2^-$.

Ejemplo 3.5
Modifiquemos ahora el ejemplo anterior.
Supongamos ahora que $w_1^+ = w_1^- = 2$ y $w_2^+ = w_2^- = 1$.

Obtendríamos:

$$\begin{aligned} & min\ 2d_1^+ + 2d_1^- + d_2^+ + d_2^- \\ & s.a: x_1 + 3x_2 \leq 24 \\ & \qquad 2x_1 + x_2 \leq 18 \\ & \qquad x_1 + x_2 \leq 10 \\ & \qquad 2x_1 + 3x_2 - d_1^+ + d_1^- = 26 \\ & \qquad 4x_1 + 2x_2 - d_2^+ + d_2^- = 20 \\ & \qquad x_i, d_i^-, d_i^+ \geq 0, i = 1, 2 \end{aligned}$$

Observamos que las ponderaciones asociadas a las variables de desviación de cada meta pueden ser distintas como en el **Ejemplo 3.4** o iguales, como en el **Ejemplo 3.5**.

3.4. Método MINMAX

Tomando como base los conceptos expuestos en [4], pasamos a desarrollar el último método de resolución de Programación por Metas que abordaremos en este trabajo.

El método MINMAX fue introducido por Flavell en 1976 y es también conocido como Programación por Metas de Chebyshev.
Este método consiste en minimizar la máxima desviación entre las metas, tomando como distancia, la distancia de Chebyshev, también denominada métrica L_∞.

El modelo general de este método se puede escribir como sigue:

$$\begin{aligned} & min\ \lambda \\ & s.a : |a_i(d_i^-, d_i^+)| \leq \lambda \\ & \quad f_i(x) + d_i^- - d_i^+ = m_i,\ i = 1, ..., p \\ & \quad x \in S \end{aligned} \tag{3.2}$$

La ventaja de este método respecto al método lexicográfico y el método ponderado es que, en lugar de alcanzar la solución óptima para las metas a las que le demos mayor prioridad o preferencia de orden, logramos obtener una solución de compromiso para todas las metas involucradas.

Ejemplo 3.6
Volvamos al **ejemplo 2.3** hallado en [4] considerando que nuestro objetivo es:

$$
\begin{aligned}
&min\ a = \lambda \\
&s.a: \frac{d_1^+}{120} \leq \lambda \\
&\quad \frac{d_2^-}{7000} \leq \lambda \\
&\quad \frac{d_3^-}{40} \leq \lambda \\
&\quad \frac{d_4^-}{40} \leq \lambda \\
&\quad 4x_1 + 3x_2 + d_1^- - d_1^+ = 120 \\
&\quad 100x_1 + 150x_2 + d_2^- - d_2^+ = 7000 \\
&\quad x_1 + d_3^- - d_3^+ = 40 \\
&\quad x_2 + d_4^- - d_4^+ = 40 \\
&\quad 2x_1 + x_2 \geq 50 \\
&\quad x_1 + x_2 \leq 75 \\
&\quad x_1, x_2 \geq 0, d_i^-, d_i^+ \geq 0, i = 1, 2, 3, 4
\end{aligned}
$$

3.5. Soluciones aportadas por los métodos y eficiencia

En los métodos considerados, las componentes del vector de logro son funciones lineales de las variables de desviación positiva y negativa respecto a las metas consideradas. Cada a_i depende de d_i^- y/o d_i^+ y, como $f_i(x)+d_i^- -d_i^+ = m_i$, es función de x. Simplificaremos entonces representando cada a_i como $a_i(x)$.

En los casos resueltos con los métodos anteriores, cada $a_i(x)$ es lineal y no negativa. En general, para intentar alcanzar los niveles establecidos para cada $f_i(x)$, las propiedades deseables para las $a_i(x)$ deben incluir que sean crecientes en las variables de desviación y que sean no negativas.

3.5.1. Método lexicográfico

Utilizando el Método Lexicográfico, supongamos que 1,. . . ,p es el orden decreciente de las prioridades asignadas por el decisor a las metas establecidas.

Supuesto que $\overline{m_1} = f_1(x^1)$, siendo x^1 solución del problema:

$$
\begin{aligned}
&min\ a_1(x) \\
&s.a: f_1(x) + d_1^- - d_1^+ = m_1 \\
&\quad x \in S
\end{aligned}
$$

en la iteración $i \in \{2, ..., p\}$ se resuelve el problema:

$$\begin{aligned} & min\ a_i(x) \\ & s.a : f_j(x) + d_j^- - d_j^+ = m_j,\ j = 1, ..., i \\ & \quad f_j(x) = \bar{m}_j, j = 1, ..., i-1 \\ & \quad x \in S \end{aligned} \tag{3.3}$$

3.5.2. Propiedad

Si existen, las soluciones de 3.3 son eficientes o débilmente eficientes respecto a las metas consideradas.

Demostración
Sea $\bar{x}$ una solución óptima de 3.3. Si $\bar{x}$ no es débilmente eficiente respecto a las metas 1,. . . , i, $\exists x' \in S$ tal que $a_j(x') < a_j(\bar{x}), j = 1, \ldots, i$. Entonces, trivialmente, $\bar{x}$ no es solución óptima de (3.3). Si $\bar{x}$ es solución óptima única de 3.3, obviamente $\nexists x' \in S$ tal que $a_j(x') \leq a_j(\bar{x})$, j=1,. . . , i y $a_{j_0}(x') < a_{j_0}(\bar{x})$, para algún $j_0 \in \{1, ..., i\}$ .Si $\bar{x}$ no es solución óptima única de 3.3, sea S_i el conjunto de soluciones de este problema.

Las soluciones del problema:

$$\begin{aligned} & min\ \frac{1}{i} \sum_{j=1}^{i} a_j(x) \\ & s.a : x \in S_i \end{aligned}$$

son soluciones de 3.3 que, también, son eficientes respecto a las metas consideradas. Por tanto, en los procesos de resolución nos podemos referir a estas últimas.

3.5.3. Método ponderado

Como se plantea la resolución del problema:

$$\begin{aligned} & min \sum_{j=1}^{p} \pi_j a_j(x) \\ & s.a : f_j(x) + d_j^- - d_j^+ = m_j, 1, ..., p \\ & \quad x \in S \end{aligned}$$

con $\pi_j > 0$, para j=1, . . . , p, las soluciones son eficientes respecto a las metas consideradas (ver, por ejemplo, ()).

3.5.4. Método MINMAX(Chebyshev)

Se plantea el problema:

$$\min_{x\in S} \max_{j\in\{1,\ldots,p\}} \pi_j a_j(x)$$

con $\pi_j > 0$, para j=1, . . . , p; es decir:

$$\begin{aligned} &min\ \lambda \\ &s.a : \pi_j a_j(x) \leq \lambda, \forall j \in \{1, ..., p\} \\ &\quad f_j(x) + d_j^- - d_j^+ = m_j,\ j = 1, ..., p \\ &\quad x \in S \end{aligned} \tag{3.4}$$

Sea $(\bar{\lambda}, \bar{x})$ solución óptima del anterior problema. Si suponemos que $\bar{x}$ no es débilmente eficiente para las metas consideradas, $\exists x' \in S$ tal que $a_j(x') < a_j(\bar{x}), j = 1, ..., p$, entonces (λ', x') es solución del citado problema anterior con $\lambda' = \text{máx}_{j\in\{1,\ldots,p\}}\, \pi_j a_j(x') < \bar{\lambda}$ y, en consecuencia, $(\bar{\lambda}, \bar{x})$ no sería solución óptima de 3.3. Por tanto, las soluciones de 3.3 son débilmente eficientes para las metas consideradas.

Además, si S_λ es el conjunto de soluciones de 3.3, la solución del problema:

$$\begin{aligned} &min\ \frac{1}{p}\sum_{j=1}^{p} \pi_j a_j(x) \\ &s.a : x \in S_\lambda \end{aligned}$$

es eficiente para las metas consideradas. Por tanto, entre las soluciones de 3.3 existe, al menos, una que es eficeinte para las metas consideradas.

Nota: Aunque cada ponderación π_j pueden formar parte de la expresión de $a_j(x)$, hemos de resaltar que su consideración, aparte de asignar pesos, implica la correción del problema 3.3 ya que hace posible la necesaria eliminación de las unidades en que se expresan las diferentes metas.

4

Aplicaciones a distintos problemas

4.1. Introducción

La dimensión habitual de los problemas de Programación por Metas y la complejidad de los cálculos necesarios para su resolución, imponen el uso del computador.
Existen programas de carácter general y programas específicos en los que se puede apoyar la aplicación de las metodologías introducidas previamente.

En este trabajo, como ilustración de las disponibilidades, para resolver distintos ejemplos, usaremos complementos de hojas de cálculo, el programa LINGO y programas específicos de paquetes de R.

4.2. Software utilizado para la resolución de problemas de Programación por Metas

4.2.1. LINGO

Estudiando los aspectos técnicos de LINGO y profundizando en su uso según [3], podemos detallar en el presente trabajo las características más destacadas de este software aplicándolo a la resolución diversos ejemplos.

LINGO (Linear Generalize Optimizer) es un software creado por Lindo System, que presenta versiones en las que no se tiene límite de variables, ni restricciones. Además, LINGO cuenta con versiones para Linux, Mac y Windows de 32 y 64 bits.

Es importante destacar que se pueden obtener versiones de prueba de LINGO en [3]. En esta misma página web podemos encontrar el manual del usuario, gracias

al que podemos aprender a modelar y resolver aquellos problemas que sean de nuestro interés.

LINGO resuelve problemas de programación lineal, no lineal, entera, cuadrática, ... Cabe resaltar, además, diversas ventajas que justifican el uso de este software ya que su lenguaje de modelado es bastante sencillo y similar al que utilizamos en la escritura de forma habitual. Esto hace que el tiempo que utilicemos para la introducción del problema sea mínimo.

Por otro lado, LINGO es capaz de obtener los datos necesarios del problema de bases de datos y hojas de cálculo. Es más, tiene la capacidad de devolver las soluciones de los problemas a las hojas de cálculo y bases de datos.

4.2.2. R y RStudio

R es un software de uso libre que presenta versiones para Windows, Linux y Mac.
R fue desarrollado por Bell Laboratories y es capaz de resolver problemas de programación lineal, no lineal, series temporales,... También se utiliza para la realización de gráficos y se pueden implementar fácilmente diversas librerías para resolver distintos problemas de optimización.

Haremos uso del mismo siguiendo las indicaciones del manual que podemos encontrar en [9].

Por otro lado, estudiamos RStudio. RStudio es un entorno desarrollado para R. Es de código abierto y comercial y está disponible para Windows, Linux y Mac.

Podemos descargarlo y encontrar más infromación de este programa en [8].

Cabe destacar, por último, los paquetes **goalprog** y **glpkAPI**, que son paquetes específicos de R para resolver los problemas de Programación por Metas, problemas en los que nos centramos a lo largo de este trabajo.

4.2.3. Complementos de optimización en hojas de cálculo (Excel de Office, Calc de LibreOffice, Spreadsheets de Google, ...)

Si se trata de problemas sencillos de Programación Lineal o Entera, una herramienta a utilizar, puede ser el correspondiente complemento de resolución que aparece en las hojas de cálculo más populares. Para nuestro trabajo usaremos Solver de Excel. Solver utiliza el método del Simplex para resolver problemas de Programación Lineal y Programación Entera.

Sin embargo, se debe tener presente que Solver presenta límite de variables y restricciones de los problemas a resolver. Esto límita su uso a problemas sencillos.

LibreOffice Calc, además de Spreadsheets de Google (entre otras hojas de cálculo libres) también presenta una herramienta de optimización muy similar a Solver de Excel.

4.3. Resolución de los problemas propuestos

Para la resolución de los ejemplos que se tratarán a continuación vamos a utilizar el complemento SOLVER para construir los correspondientes modelos de Programación Lineal, además utilizaremos LINGO y R. Cabe resaltar que la resolución que haremos en Solver (Excel) es similar si se trabaja con complementos de otras hojas de cálculo.

Los correspondientes paquetes de R se utilizarán siguiendo las pautas en los respectivos manuales de referencia. Obviamente, como trabajamos en la práctica con problemas de Programación Lineal, podemos utilizar cualquier paquete conveniente de R que resuelva este tipo de problemas.

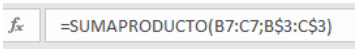
4.3.1. Resolución del Ejemplo 1.1

En primer lugar resolveremos el **Ejemplo1.1** ayundandonos del complemento Solver de Excel. Este complemento se debe cargar previamente y se localiza en la pestaña de datos de la hoja de cálculo.
Lo primero que debemos hacer es introducir los datos del modelo en una hoja de Excel, como se muestra a continuación.

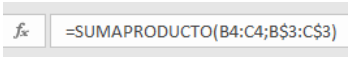
	A	B	C	D	E	F
1						
2		x_1	x_2	Total		
3						
4		1	3	0		24
5		2	1	0		18
6		1	1	0		10
7	Coef. Objeti	-2	1	0		

En una columna introducimos los coeficientes de cada restricción y de la función objetivo de x_1, en otra columna lo ánalogo con x_2.

La función objetivo introducida (en este caso en la casilla D7) es:



Las funciones que se introducen como total (exceptuando la función objetivo) hace referencia al primer miembro de las restricciones. Por ejemplo, en la casilla D8 se ha introducido $x_1 + 3x_2$, como sigue:



Y en la última columna (en este ejemplo la F) se introduce el segundo miembro de las restricciones.

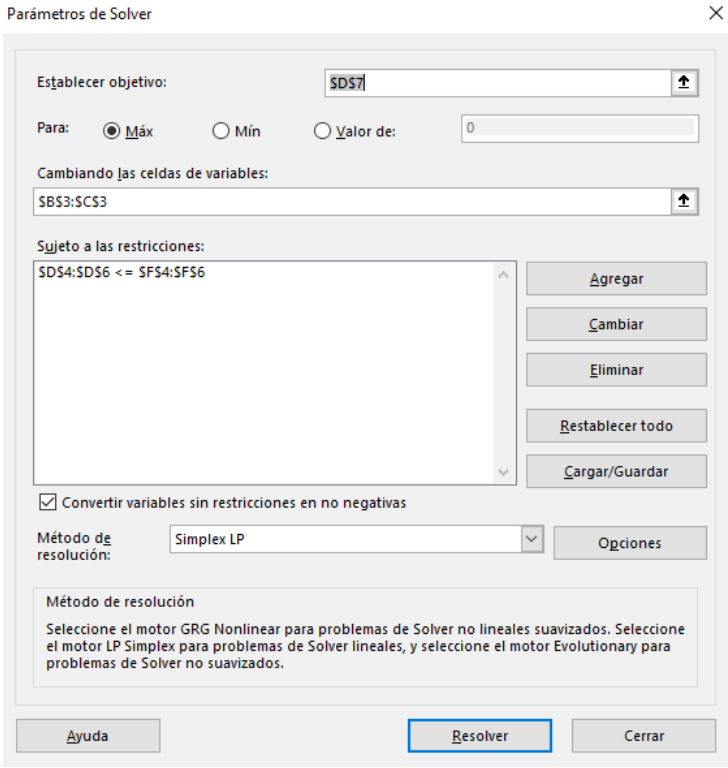
Una vez introducido el modelo en la hoja de cálculo, debemos completar los campos que se nos solicitan en la ventana que nos aparece al abrir Solver.

Establecemos el objetivo, es decir, indicamos la casilla en la que hemos introducido la función objetivo (D7 para este ejemplo).
Además debemos indicar las variables, es decir x_1 y x_2

Debemos ahora introducir las restricciones:

- La columna de totales deben ser menor o igual que la columna donde hemos introducido los valores del segundo miembro de las restricciones.

Siguiendo estos pasos, la ventana de Solver nos quedaría como sigue:

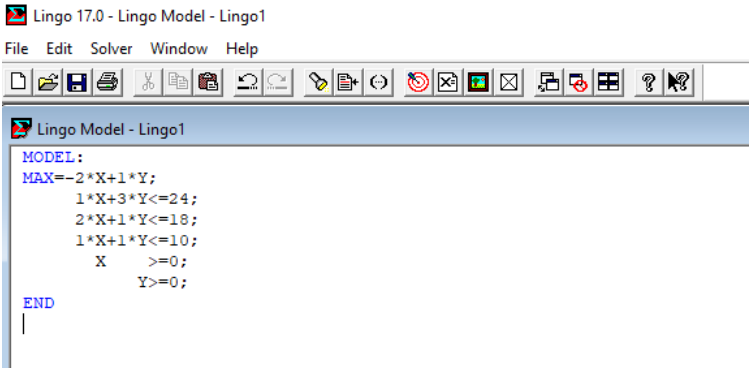


Al resolver el problema obtenemos la siguiente solución:

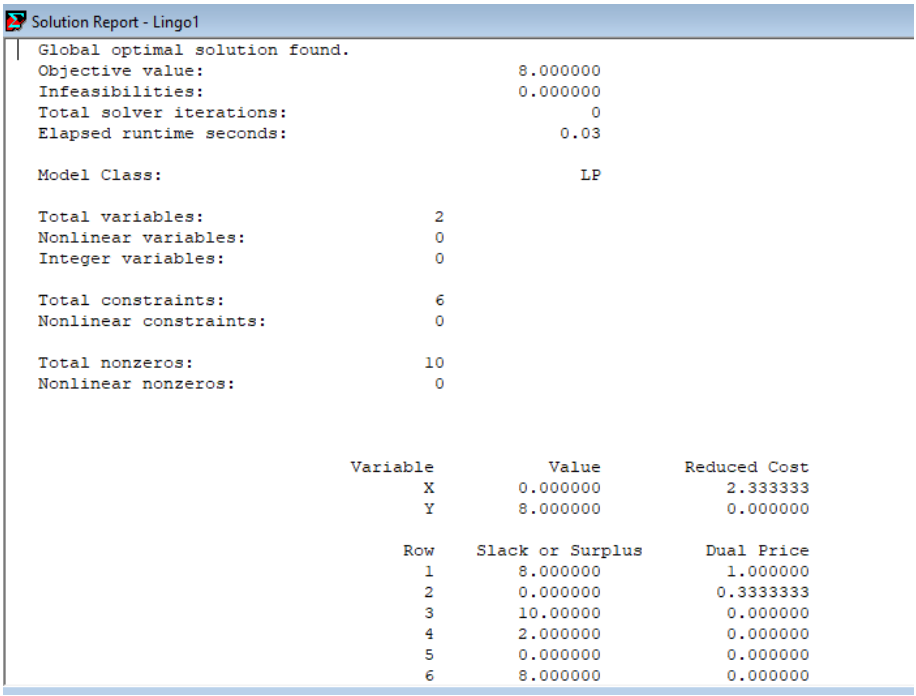
	A	B	C	D	E	F
1						
2		x_1	x_2	Total		
3		0	8			
4		1	3	24		24
5		2	1	8		18
6		1	1	8		10
7	Coef. Objeti	-2	1	8		

Por tanto, obtenemos $x_1 = 0$, $x_2 = 8$ y la función objetivo valdría 8. Vemos que esta solución es óptima.

Pasemos ahora a resolver el problema con el Software LINGO. Introducimos el modelo:



Vemos que la escritura del modelo es muy intuitiva y similar a la escritura diaria. Ejecutando el modelo, obtenemos como resultado:



Misma solución que la obtenida anteriormente (con Solver).

En este momento pasamos a resolver este mismo ejemplo usando RStudio. Implementando así el código siguiente:

```
library(linprog)
z <- c(-2,1)
A <- matrix(c(1,2,1,3,1,1),ncol=2)
b <- c(24,18,10)
dir <- rep("<=",3)
solopt <- solveLP(z,b,A,maximum=TRUE, dir)
summary(solopt)
```

Cargamos la librería linprog (necesaria para la resolución del programa). En el vector z introducimos los coeficientes de la función objetivo y mediante la matriz A introducimos los coeficientes de las restricciones. También debemos añadir un vector b con el valor de las restricciones y por último guardamos la solución del problema en solopt y la mostramos por pantalla.

Obtenemos como solución:

```
Results of Linear Programming / Linear Optimization

Objective function (Maximum): 8

Solution
  opt
1   0
2   8
```

Podemos observar que con los tres software hemos logrado obtener la misma solución. Como se indica al final del capítulo 1, esta solución es eficiente para el problema multiobjetivo.

4.3.2. Resolución del Ejemplo 3.2

Veamos un ejemplo en el que empleamos paquetes de RStudio distintos de los usados anteriormente.

las instrucciones correspondientes son:

```
library(lpSolve)
library(linprog)

z <- c(0,0,1,0,0,0,0,0,0,0)
A <- matrix(c(7,6,1,-1,0,0,0,0,0,0,
              2,3,0,0,1,-1,0,0,0,0,
              6,5,0,0,0,0,0,1,-1,0,0,
              0,1,0,0,0,0,0,0,0,1,-1),nrow=4, byrow = TRUE)
b <- c(30,12,30,7)
dir <- rep("=",4)
sal<-lp("min",z,A,dir,b)
sal$solution
#-----------------------------------------------------------------------
z <- c(0,0,0,0,0,1,0,0,0,0)
A <- matrix(c(7,6,-1,1,0,0,0,0,0,0,
              2,3,0,0,-1,1,0,0,0,0,
              6,5,0,0,0,0,-1,1,0,0,
              0,1,0,0,0,0,0,0,-1,1,
              0,0,0,1,0,0,0,0,0,0),nrow=5, byrow = TRUE)
b <- c(30,12,30,7,0)
dir <- rep("=",5)
sal<-lp("min",z,A,dir,b)
sal$solution
#-----------------------------------------------------------------------
z <- c(0,0,0,0,0,0,1,0,0,0)
A <- matrix(c(7,6,-1,1,0,0,0,0,0,0,
              2,3,0,0,-1,1,0,0,0,0,
              6,5,0,0,0,0,-1,1,0,0,
              0,1,0,0,0,0,0,0,-1,1,
              0,0,0,1,0,0,0,0,0,0,
              0,0,0,0,0,1,0,0,0,0),nrow=6, byrow = TRUE)
b <- c(30,12,30,7,0,0)
dir <- rep("=",6)
sal<-lp("min",z,A,dir,b)
sal$solution
#-----------------------------------------------------------------------
z <- c(0,0,0,0,0,0,0,0,0,1)
A <- matrix(c(7,6,-1,1,0,0,0,0,0,0,
              2,3,0,0,-1,1,0,0,0,0,
              6,5,0,0,0,0,-1,1,0,0,
              0,1,0,0,0,0,0,0,-1,1,
              0,0,0,1,0,0,0,0,0,0,
              0,0,0,0,0,1,0,0,0,0,
              0,0,0,0,0,0,1,0,0,0),nrow=7, byrow = TRUE)
b <- c(30,12,30,7,0,0,0)
dir <- rep("=",7)
sal<-lp("min",z,A,dir,b)
sal$solution
```

En el código anterior se resuelve el problema con el método secuencial. Por tanto, hemos de resolver 4 problemas independientes para resolver la totalidad del problema que nos ocupa.

Todos los problemas se resuelven de manera ánaloga. Lo primero que se hace es cargar las librerías a utilizar (en este caso lpSolve y linprog). Creamos un vector z para los coeficientes de la función objetivo, una matriz para los coeficientes del primer miembro de las restricciones y otro vector para el segundo miembro de las restricciones.

Además introducimos en dir el tipo de desigualdades o en su caso igualdad de las restricciones que tenemos. Por último, resolvemos el problema mediante el comando lp y guardamos la solución (en este caso la hemos denominado sal) e imprimimos por pantalla la misma, la cuál es:

```
> sal$solution
 [1] 0 6 6 0 6 0 0 0 0 1
```

Este mismo problema se puede resolver de manera más escueta gracias al paquete de RStudio: **goalprog**. Veamos el código en el que se procede a su resolución.

```
1  library(lpsolve)
2  library(linprog)
3  library(goalprog)
4  coefficients <- matrix( c( 7, 6,2,3,6,5,0,1), nrow=4, byrow=TRUE )
5  targets <- c( 30,12,30,7 )
6  achievements <- data.frame( matrix(
7    c( 1, 1, 0, 1,
8       2, 2, 0, 1,
9       3, 3, 1, 0,
10      4, 4, 0, 1), nrow=4, byrow=TRUE ) )
11 names( achievements ) <- c( "objective", "priority", "p", "n" )
12 soln <- llgp(coefficients, targets, achievements)
13 soln
```

Inicialmente cargamos las librerías necesarias (lpSolve, linprog y goalprog) y creamos:

- Una matriz que denominaremos **coefficients** en la que introduciremos los coeficientes del primer miembro de las restcciones del problema.
- Un vector denominado **targets** en el que introducimos el valor del segundo miembro de las restricciones.
- Una matriz llamada **achievements** en la que introducimos en la primera columna el número de la restricción, en la segunda columna la prioridad que le demos a esa restrcción, en la tercera columna el coeficiente de la variable de

desviación positiva de la correspondiente restricción y en la cuarta columna se introduce el coeficiente de la correspondiente variable de desviación negativa.
- Identificamos cada columna de la matriz anterior.
- Resolvemos el problema mediante el comando llgp y guardamos la solución (en este caso la hemos denominado soln).
- Imprimimos por pantalla la solución, como se muestra a continuación:

```
Current Solution
             Value
P2    6.000000e+00
X2    6.000000e+00
P1    6.000000e+00
N4    1.000000e+00
```

Vemos que con ambos códigos se obtiene la misma solución y observamos que esta es óptima.

Sin embargo, se puede observar que utilizando el paquete **goalprog** el código es mucho más simple y escueto. Esto es positivo para la resolución del problema pues debido a que el código es más corto, hay menos posibilidad de error en la escritura.

4.3.3. Resolución del Ejemplo 3.3

A continuación, resolvemos el **ejemplo 3.3** con el paquete lpSolveAPI de RStudio mediante el siguiente código.

```
library(lpSolveAPI)
 lgp <- make.lp(0, 10)
set.objfn(lgp, c(0,0,0,0,1,0,0,0,0,0))
#min d2-
add.constraint(lgp, c(2,1,0,0,0,0,0,0,0,0), ">=", 50)
add.constraint(lgp, c(1,1,0,0,0,0,0,0,0,0), "<=", 75)
add.constraint(lgp, c(4,3,1,-1,0,0,0,0,0,0), "=", 120)
add.constraint(lgp, c(100,150,0,0,1,-1,0,0,0,0), "=", 7000)
add.constraint(lgp, c(1,0,0,0,0,0,1,-1,0,0), "=", 40)
add.constraint(lgp, c(0,1,0,0,0,0,0,0,1,-1), "=", 40)
lgp
solve(lgp)
get.variables(lgp)
get.objective(lgp)
#min (d3-) + (d4-)
add.constraint(lgp,c(0,0,0,0,1,0,0,0,0,0),"=",0)
set.objfn(lgp, c(0,0,0,0,0,0,1,0,1,0))
solve(lgp)
get.variables(lgp)
get.objective(lgp)
# min d1+
add.constraint(lgp,c(0,0,0,0,0,0,1,0,1,0),"=",5)
set.objfn(lgp, c(0,0,0,1,0,0,0,0,0,0))
lgp
solve(lgp)
get.variables(lgp)
get.objective(lgp)
get.constraints(lgp)
get.constr.value(lgp)
```

Lo que hacemos es:

- Creamos un nuevo problema con 0 restricciones y 10 variables.
- Añadimos la función objetivo con set.objfn
- Añadimos las restricciones mediante add. constraint
- Obtenemos la solución con solve(lgp)
- Obtenemos el valor de las variables y de la función objetivo
- Análogamente resolvemos los problemas restantes (asociados a cada componente del vector de logro) para la resolución total del problema

Al ejecutar el código obtenemos:

```
> lgp
Model name:
  a linear program with 10 decision variables and 8 constraints
> solve(lgp)
[1] 0
> get.variables(lgp)
 [1]   35   40    0  140    0 2500    5    0    0    0
> get.objective(lgp)
[1] 140
> get.constraints(lgp)
[1]  110   75  120 7000   40   40    0    5
> get.constr.value(lgp)
[1]   50   75  120 7000   40   40    0    5
```

Efectivamente hemos hallado una solución óptima.

4.4. Otra aplicación: Regresión Lineal y Programación por Metas

Es conocido que, dado un conjunto de n pares (x_i, y_i) de la variable numérica bidimensional (X,Y), la regresión lineal trata de ajustar, a la correspondiente nube de puntos, la recta y=ax+b, siendo a y b parámetros que hay que determinar.

Sabemos que el cálculo de dichos parámetros, cuando se minimizan las desviaciones entre los valores y_i y los correspondientes $ax_i + b$, permite construir la recta de regresión de Y sobre X. Dicho cálculo se realiza minimizando la suma de los cuadrados de $y_i - (ax_i + b)$ (método de mínimos cuadrados).

Alternativamente, para hacer el citado ajuste, podríamos utilizar la Programación por Metas planteando el siguiente problema:

$$\begin{aligned} min\ & \sum_{i=1}^{n}(d_i^- + d_i^+) \\ s.a: & ax_i + b + d_i^- - d_i^+ = y_i,\ 1, ..., n \\ & d_i^-, d_i^+ \geq 0,\ i = 1, ..., n \end{aligned}$$

En este problema, las variables son a, b, d_i^- y d_i^+. Como, en principio, a y b pueden ser negativas o no negativas, para resolver el problema anterior como un problema de Programación Lineal, cambiamos:

$$\begin{aligned} a &= a^+ - a^-,\ a^+,\ a^- \geq 0 \\ b &= b^+ - b^-,\ b^+,\ b^- \geq 0 \end{aligned}$$

y el modelo anterior se convierte en:

$$
\begin{aligned}
&min \sum_{i=1}^{n}(d_i^- + d_i^+) \\
&s.a : (a^+ - a^-)x_i + (b^+ - b^-) + d_i^- - d_i^+ = y_i,\ 1, ..., n \\
&\quad a^+, a^-, b^+, b^-, d_i^-, d_i^+ \geq 0
\end{aligned}
$$

Usando convenientemente la solución se encuentra una recta de regresión de Y sobre X. De forma similar se podría determinar una recta de regresión de X sobre Y.

4.5. Resolución de otros ejemplos

Ejemplo 2.1

Vemos que al resolver el correspondiente problema tanto con Solver como con RStudio obtenemos la misma solución: $x_1 = 2.4,\ x_2 = 7.2,\ d_1^+ = d_2^+ = d_2^- = 0\ y\ d_1^- = 1.6$.

	A	B	C	D	E	F	G	H
1	x_1	x_2	d_1^+	d_1^-	d_2^+	d_2^-	Total	
2	2,4	7,2	0	1,6	0	0		
3								
4	1	3	0	0	0	0	24	24
5	2	1	0	0	0	0	12	18
6	1	1	0	0	0	0	9,6	10
7	2	3	-1	1	0	0	28	28
8	4	2	0	0	-1	1	24	24
9	0	0	1	1	1	1	1,6	

```
#Example 2.1.
library(lpSolve)
library(linprog)
z <- c(0,0,1,1,1,1)
A <- matrix(c(1,3,0,0,0,0,
              2,1,0,0,0,0,
              1,1,0,0,0,0,
              2,3,-1,1,0,0,
              4,2,0,0,-1,1),nrow=5, byrow = TRUE)
A
b <- c(24,18,10,28,24)
const.dir=c("<=","<=","<=","=","=")
sal<-lp("min",z,A,const.dir,b)
sal
sal$solution
```

Ejemplo 2.2

Vemos que al resolver el correspondiente problema tanto con Solver como con RStudio obtenemos la misma solución: $x_1 = 0,\ x_2 = 6,\ d_1^+ = 6,\ d_2^+ = 6,\ d_4^- = 1$ *y* $d_1^- = d_2^- = d_3^+ = d_3^- = d_4^+ = 0$.

	A	B	C	D	E	F	G	H	I	J	K	L
1	x_1	x_2	d_1^+	d_1^-	d_2^+	d_2^-	d_3^+	d_3^-	d_4^+	d_4^-	Total	
2	0	6	6	0	6	0	0	0	0	1		
3	7	6	-1	1	0	0	0	0	0	0	30	30
4	2	3	0	0	-1	1	0	0	0	0	12	12
5	6	5	0	0	0	0	-1	1	0	0	30	30
6	0	1	0	0	0	0	0	0	-1	1	7	7
7	0	0	0	1	0	1	1	0	0	1	1	

```
#EXAMPLE 2.2
library(lpSolve)
library(linprog)
z <- c(0,0,1,0,1,0,0,1,1,0)
A <- matrix(c(7,6,1,-1,0,0,0,0,0,0,
              2,3,0,0,1,-1,0,0,0,0,
              6,5,0,0,0,0,1,-1,0,0,
              0,1,0,0,0,0,0,0,1,-1),nrow=4, byrow = TRUE)
A
b <- c(30,12,30,7)
dir <- rep("=",4)
sal<-lp("min",z,A,dir,b)
sal
sal$solution
```

Ejemplo 2.3

Vemos que al resolver el correspondiente problema tanto con Solver como con RStudio obtenemos la misma solución: $x_1 = 2.5,\ x_2 = 45,\ d_1^+ = 25,\ d_3^- = 37.5,\ d_4^+ = 5$ *y* $d_1^- = d_2^- = d_2^+ = d_3^+ = d_4^- = 0$.

	A	B	C	D	E	F	G	H	I	J	K	L
1	x_1	x_2	d_1^+	d_1^-	d_2^+	d_2^-	d_3^+	d_3^-	d_4^+	d_4^-	Total	
2	2,5	45	25	0	0	0	0	37,5	5	0		
3	4	3	-1	1	0	0	0	0	0	0	120	120
4	100	150	0	0	-1	1	0	0	0	0	7000	7000
5	1	0	0	0	0	0	-1	1	0	0	40	40
6	0	1	0	0	0	0	0	0	-1	1	40	40
7	2	1	0	0	0	0	0	0	0	0	50	50
8	1	1	0	0	0	0	0	0	0	0	47,5	75
9	0	0	1	0	0	1	0	1	0	1	62,5	

```r
#Example 2.3.
library(lpSolve)
library(linprog)
z <- c(0,0,1,0,0,1,0,1,0,1)
A <- matrix(c(4,3,-1,1,0,0,0,0,0,0,
              100,150,0,0,-1,1,0,0,0,0,
              1,0,0,0,0,0,-1,1,0,0,
              0,1,0,0,0,0,0,0,-1,1,
              2,1,0,0,0,0,0,0,0,0,
              1,1,0,0,0,0,0,0,0,0),nrow=6, byrow = TRUE)
A
b <- c(120,7000,40,40,50,75)
const.dir=c("=","=","=","=",">=","<=")
sal<-lp("min",z,A,const.dir,b)
sal
sal$solution
```

Ejemplo 3.1

Vemos que al resolver el correspondiente problema tanto con Solver como con RStudio obtenemos la misma solución: $x_1 = 0,\ x_2 = 7,\ d_1^- = 3,\ d_2^- = 11,\ d_3^- = 3,\ d_5^+ = 2$ y $d_1^+ = d_2^+ = d_3^+ = d_4^- = d_4^+ = d_5^- = 0$.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1														
2	x_1	x_2	d_1^+	d_1^-	d_2^+	d_2^-	d_3^+	d_3^-	d_4^+	d_4^-	d_5^+	d_5^-	Total	
3	3	7	0	0	0	5	0	0	6	0	14	0		
4	1	3	-1	1	0	0	0	0	0	0	0	0	24	24
5	2	1	0	0	-1	1	0	0	0	0	0	0	18	18
6	1	1	0	0	0	0	-1	1	0	0	0	0	10	10
7	2	3	0	0	0	0	0	0	-1	1	0	0	21	21
8	4	2	0	0	0	0	0	0	0	0	-1	1	12	12
9	0	0	1	0	1	0	1	0	0	0	0	0	0	
10														
11														
12														
13														
14	x_1	x_2	d_1^+	d_1^-	d_2^+	d_2^-	d_3^+	d_3^-	d_4^+	d_4^-	d_5^+	d_5^-	Total	
15	3	7	0	0	0	5	0	0	6	0	14	0		
16	1	3	-1	1	0	0	0	0	0	0	0	0	24	24
17	2	1	0	0	-1	1	0	0	0	0	0	0	18	18
18	1	1	0	0	0	0	-1	1	0	0	0	0	10	10
19	2	3	0	0	0	0	0	0	-1	1	0	0	21	21
20	4	2	0	0	0	0	0	0	0	0	-1	1	12	12
21	0	0	1	0	1	0	1	0	0	0	0	0	0	0
22	0	0	0	0	0	0	0	0	0	1	0	0	0	
23														
24														
25														
26	x_1	x_2	d_1^+	d_1^-	d_2^+	d_2^-	d_3^+	d_3^-	d_4^+	d_4^-	d_5^+	d_5^-	Total	
27	0	7	0	3	0	11	0	3	0	0	2	0		
28	1	3	-1	1	0	0	0	0	0	0	0	0	24	24
29	2	1	0	0	-1	1	0	0	0	0	0	0	18	18
30	1	1	0	0	0	0	-1	1	0	0	0	0	10	10
31	2	3	0	0	0	0	0	0	-1	1	0	0	21	21
32	4	2	0	0	0	0	0	0	0	0	-1	1	12	12
33	0	0	1	0	1	0	1	0	0	0	0	0	0	0
34	0	0	0	0	0	0	0	0	0	1	0	0	0	0
35	0	0	0	0	0	0	0	0	0	0	1	0	2	

```
#EXAMPLE 3.1.
library(lpSolve)
library(linprog)
z <- c(0,0,1,0,1,0,1,0,0,0,0,0)
A <- matrix(c(1,3,-1,1,0,0,0,0,0,0,0,0,
              2,1,0,0,-1,1,0,0,0,0,0,0,
              1,1,0,0,0,0,-1,1,0,0,0,0,
              2,3,0,0,0,0,0,0,-1,1,0,0,
              4,2,0,0,0,0,0,0,0,0,-1,1),nrow=5, byrow = TRUE)
A
b <- c(24,18,10,21,12)
dir <- rep("=",5)
sal<-lp("min",z,A,dir,b)
sal
sal$solution

library(lpSolve)
library(linprog)
z <- c(0,0,0,0,0,0,0,0,0,1,0,0)
A <- matrix(c(1,3,-1,1,0,0,0,0,0,0,0,0,
              2,1,0,0,-1,1,0,0,0,0,0,0,
              1,1,0,0,0,0,-1,1,0,0,0,0,
              2,3,0,0,0,0,0,0,-1,1,0,0,
              4,2,0,0,0,0,0,0,0,0,-1,1,
              0,0,1,0,1,0,1,0,0,0,0,0),nrow=6, byrow = TRUE)
A
b <- c(24,18,10,21,12,0)
dir <- rep("=",6)
sal<-lp("min",z,A,dir,b)
sal
sal$solution


library(lpSolve)
library(linprog)
z <- c(0,0,0,0,0,0,0,0,0,0,1,0)
A <- matrix(c(1,3,-1,1,0,0,0,0,0,0,0,0,
              2,1,0,0,-1,1,0,0,0,0,0,0,
              1,1,0,0,0,0,-1,1,0,0,0,0,
              2,3,0,0,0,0,0,0,-1,1,0,0,
              4,2,0,0,0,0,0,0,0,0,-1,1,
              0,0,1,0,1,0,1,0,0,0,0,0,
              0,0,0,0,0,0,0,0,0,0,1,0,0),nrow=7, byrow = TRUE)
A
b <- c(24,18,10,21,12,0,0)
dir <- rep("=",7)
sal<-lp("min",z,A,dir,b)
sal
sal$solution
```

Ejemplo 3.4

Vemos que al resolver el correspondiente problema tanto con Solver como con RStudio obtenemos la misma solución: $x_1 = 1.2$, $x_2 = 7.6$, $d_1^- = 0.8$ y $d_1^+ = d_2^+ = d_2^- = 0$.

	A	B	C	D	E	F	G	H
1	x_1	x_2	d_1^+	d_1^-	d_2^+	d_2^-	Total	
2	1,2	7,6	0	0,8	0	0		
3	1	3	0	0	0	0	24	24
4	2	1	0	0	0	0	10	18
5	1	1	0	0	0	0	8,8	10
6	2	3	-1	1	0	0	26	26
7	4	2	0	0	-1	1	20	20
8	0	0	4	1	1	2	0,8	

```
library(lpSolve)
library(linprog)
z <- c(0,0,1,4,2,1)
A <- matrix(c(1,3,0,0,0,0,
              2,1,0,0,0,0,
              1,1,0,0,0,0,
              2,3,1,-1,0,0,
              4,2,0,0,1,-1),nrow=5, byrow = TRUE)
A
b <- c(24,18,10,26,20)
const.dir=c("<=","<=","<=","=","=")
sal<-lp("min",z,A,const.dir,b)
sal
sal$solution
```

Ejemplo 3.5

Vemos que al resolver el correspondiente problema tanto con Solver como con RStudio obtenemos la misma solución: $x_1 = 1.2$, $x_2 = 7.6$, $d_1^- = 0.8$ y $d_1^+ = d_2^+ = d_2^- = 0$.

	A	B	C	D	E	F	G	H
1	x_1	x_2	d_1^+	d_1^-	d_2^+	d_2^-	Total	
2	1,2	7,6	0	0,8	0	0		
3	1	3	0	0	0	0	24	24
4	2	1	0	0	0	0	10	18
5	1	1	0	0	0	0	8,8	10
6	2	3	-1	1	0	0	26	26
7	4	2	0	0	-1	1	20	20
8	0	0	2	2	1	1	1,6	

```
library(lpSolve)
library(linprog)
z <- c(0,0,2,2,1,1)
A <- matrix(c(1,3,0,0,0,0,
              2,1,0,0,0,0,
              1,1,0,0,0,0,
              2,3,1,-1,0,0,
              4,2,0,0,1,-1),nrow=5, byrow = TRUE)
A
b <- c(24,18,10,26,20)
const.dir=c("<=","<=","<=","=","=")
sal<-lp("min",z,A,const.dir,b)
sal
sal$solution
```

Ejemplo 3.6

Vemos que al resolver el correspondiente problema tanto con Solver como con RStudio obtenemos la misma solución: $\lambda = 0.4,\ x_1 = x_2 = 24,\ d_1^+ = 48,\ d_2^- = 1000,\ d_3^- = d_4^- = 16\ y\ d_1^- = d_2^+ = d_3^+ = d_4^+ = 0.$

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	λ	x_1	x_2	d_1^+	d_1^-	d_2^+	d_2^-	d_3^+	d_3^-	d_4^+	d_4^-	Total	
2	0,4	24	24	48	0	0	1000	0	16	0	16		
3	-1	0	0	0,008333333	0	0	0	0	0	0	0	1,16573E-15	0
4	-1	0	0	0	0	0	0,000142857	0	0	0	0	-0,257142857	0
5	-1	0	0	0	0	0	0	0	0,03	0	0	-3,33067E-16	0
6	-1	0	0	0	0	0	0	0	0	0	0,03	-2,94209E-15	0
7	0	4	3	-1	1	0	0	0	0	0	0	120	120
8	0	100	150	0	0	-1	1	0	0	0	0	7000	7000
9	0	1	0	0	0	0	0	-1	1	0	0	40	40
10	0	0	1	0	0	0	0	0	0	-1	1	40	40
11	0	2	1	0	0	0	0	0	0	0	0	72	50
12	0	1	1	0	0	0	0	0	0	0	0	48	75
13	1	0	0	0	0	0	0	0	0	0	0	0,4	

```
library(lpSolve)
library(linprog)
z <- c(1,0,0,0,0,0,0,0,0,0,0)
A <- matrix(c(-1,0,0,0,1/120,0,0,0,0,0,0,
              -1,0,0,0,0,1/7000,0,0,0,0,0,
              -1,0,0,0,0,0,0,0,1/40,0,0,0,
              -1,0,0,0,0,0,0,0,0,0,1/40,0,
              0,4,3,1,-1,0,0,0,0,0,0,
              0,100,150,0,0,1,-1,0,0,0,0,
              0,1,0,0,0,0,0,1,-1,0,0,
              0,0,1,0,0,0,0,0,0,1,-1,
              0,2,1,0,0,0,0,0,0,0,0,
              0,1,1,0,0,0,0,0,0,0,0),nrow=10, byrow = TRUE)
A
b <- c(0,0,0,0,120,7000,40,40,50,75)
const.dir=c("<=","<=","<=","<=","=","=","=","=",">=","<=")
sal<-lp("min",z,A,const.dir,b)
sal
sal$solution
```

4.6. Ejemplo de Regresión Lineal y Programación por Metas

Ahora se muestra un ejemplo de la regresión lineal como aplicación de la Programación por Metas.

En este problema se pretende aproximar una nube de puntos a una recta mediante regresión lineal. La nube de puntos a aproximar es:

- (1,4)
- (2,3)
- (5,5)
- (7,7)
- (9,9)
- (11,10)

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
1								$a^+x_i - a^-x_i + b^+ - b^- + d_i^- - d_i^+ = y_i$											
2																			
3	a^+	a^-	b^+	b^-	d_1^-	d_1^+	d_2^-	d_2^+	d_3^-	d_3^+	d_4^-	d_4^+	d_5^-	d_5^+	d_6^-	d_6^+			
4	0,8	0	1,4	0	1,8	0	0	0	0	0,4	0	0	0,4	0	0	0,2			
5	1	-1	1	-1	1	-1											4	4	
6	2	-2	1	-1			1	-1									3	3	
7	5	-5	1	-1					1	-1							5	5	
8	7	-7	1	-1							1	-1					7	7	
9	9	-9	1	-1									1	-1			9	9	
10	11	-11	1	-1											1	-1	10	10	
11					1	1	1	1	1	1	1	1	1	1			2,6		
12																			

Se obtiene la solución:

$$a^+ = 0.8,\ b^+ = 1.4,\ d_1^- = 1.8,\ d_3^+ = 0.4,\ d_5^- = 0.4,\ d_6^+ = 0.2$$
$$a^- = b^- = d_1^+ = d_2^- = d_2^+ = d_3^- = d_4^- = d_4^+ = d_5^+ = d_6^- = 0$$

Bibliografía

[1] Barry, R., Ralph, M.S. y Michael, E. H.(2006). *Métodos cuantitativos para los negocios.* Pearson Education.

[2] García, A. M. (1998). *Programación estocástica por metas. Teoría y aplicaciones económicas.* Tesis Doctoral. Universidad Complutense de Madrid.

[3] LINGO and optimitation modeling. *https://www.lindo.com/index.php/products/lingo-and-optimization-modeling*

[4] Jones, D. y Tamiz, M. (2010). *Practical Goal Programming.* Springer.

[5] Ríos, S., Mateos, A., Bielza, M.C. y Jiménez, A. (2004). *Investigación operativa: modelos determinísticos y estocásticos.*

[6] Romero, C. (1993). *Teoría de la decisión multicriterio.* Alianza Editorial.

[7] Romero, C. (2002). *Toma de decisiones con criterios múltiples.* Revista Electrónica de Comunicaciones y Trabajos de ASEPUMA número 1, pp 75-87.

[8] Rstudio *https://www.rstudio.com/products/rstudio/*

[9] The R manuals *https://cran.r-project.org/manuals.html*

Goal Programming

Sección de Matemáticas
Universidad de La Laguna

María Jesús Álvarez Rodríguez
Facultad de Ciencias · Sección de Matemáticas
Universidad de La Laguna
alu0100819625@ull.edu.es

Abstract

This project will focus on the study of Goal Programming within Multiobjective Programming. After having outlined the historical aspects, a general model is presented and some examples are modelled. General methods are also presented which will allow us to solve the problems of Goal Programming. Finally, the proposed problems will be solved with the software available for Linear Programming.

1. Introduction

Problematic situations, which arise in different contexts that involve people and nature, need to be resolved in an optimum manner. Some of these situations can be drawn up as mathematical models whose study is the task of Mathematical Programming or Optimization, included within Operations Research. In search for optimal solutions to these problems more than one criterion might intervene. The cases in which we have multiple criteria are included in the Multi-criteria Optimization. These criteria might conflict with each other. In some problems, the criteria that govern the optimization process impose the overcoming of certain goals. When goals that must be met are specified, we place ourselves in what is known as Goal Programming.

2. Outline of the first Chapter

In the first chapter a brief introduction will be made to the problems of Multiobjective Programming, defining some basic concepts and the general model of them. The Multicriteria problems will also be illustrated with an example.

3. Outline of the second Chapter

In this chapter basic concepts of Goal Programming, the most relevant historical aspects, the general model and some examples that will later be solved with different programmes will be discussed.

The general model of a Multicriteria Programming problem is given in the following way:

$$\begin{aligned} & min\ a_i(d_i^-, d_i^+) \\ & s.a: f_i(x) + d_i^- - d_i^+ = m_i \\ & \quad f_j(x) = f(x^j),\ j = 1, ..., i-1 \\ & \quad x \in S \end{aligned}$$

4. Outline of the third Chapter

Different methods to solve Goal Programming problems will be studied addressing the treatment of deviation variables. The following methods are distinguished:

1. Lexographic goal methods.
   In this method the goal are ordered in descending order of proiority by the decision-maker
2. Weighted goal methods.
   In this method the decison-maker assigns weights to the different a_i (components of the achievement vector)
3. MINMAX or Chebyshev methods.
   This method consists of minimising the maximum deviation between the goals, taking as distance, the distance of Chebyshev, also called metric L_∞.

5. Outline of the fourth Chapter

After making a brief introduction to the different software used in the resolution of Goal Programming problems, we will focus on solving the problems that have been proposed throughout this study. Primarily, the problems will be solved with Excel Solver plugin and the free Rstudio software.

References

[1] BARRY, R., RALPH, M.S. Y MICHAEL, E. H.(2006). *Métodos cuantitativos para los negocios.* Pearson Education.

[2] GARCÍA, A. M. (1998). *Programación estocástica por metas. Teoría y aplicaciones económicas.* Tesis Doctoral. Universidad Complutense de Madrid.

[3] LINGO AND OPTIMITATION MODELING. *https://www.lindo.com/index.php/products/lingo-and-optimization-modeling*

[4] JONES, D. Y TAMIZ, M. (2010). *Practical Goal Programming.* Springer.

[5] RÍOS, S., MATEOS, A., BIELZA, M.C. Y JIMÉNEZ, A. (2004). *Investigación operativa: modelos determinísticos y estocásticos.*

[6] ROMERO, C. (1993). *Teoría de la decisión multicriterio.* Alianza Editorial.

[7] ROMERO, C. (2002). *Toma de decisiones con criterios múltiples.* Revista Electrónica de Comunicaciones y Trabajos de ASEPUMA número 1, pp 75-87.

[8] RSTUDIO *https://www.rstudio.com/products/rstudio/*

[9] THE R MANUALS *https://cran.r-project.org/manuals.html*